Remarks

The specification has been amended to supply serial numbers, indicate current status and delete attorney docket numbers for the applications referenced on pages 1-7, as well as to correct capitalization errors. The abstract has been shortened to no more than 150 words.

Claims 4 and 14 have been amended to correct typographical errors.

Claims 1-26 stand rejected as being unpatentable over Leymann et al. (EP 0 817 019 A2) in view of Denny et al. (US 6,330,686 B1) (paper no. 5, page 3). This rejection is respectfully traversed.

All of the claims under rejection are based upon one of three claims: 1, 13 and 20. These claims are directed to a method, apparatus and program storage device, respectively, for implementing a shared message queue using a list structure. Referring to Fig. 2, as recited in these claims, a list (202) is defined comprising a sequence of list entries (204), each of which corresponds to a message in the queue and has an associated list entry key (LEK). Each list entry key corresponding to an uncommitted message falls within an uncommitted key range defining an uncommitted portion (208) of the list, while each list entry key corresponding to a committed message falls within a committed key range defining a committed portion (210) of the list. In response to a request to write a message to the queue, a list entry having a list entry key within the uncommitted key range is added to the list.

In applicants claimed invention, therefore, the list entry key determines whether a list entry is in the uncommitted portion or the committed portion of the list. This allows a message to be committed to the queue merely by modifying the key for the associated list entry to fall within the committed key range, moving the list entry to the committed portion of the list (as recited in claims 3, 14 and 21)). Further, this feature of applicants' claimed invention is not taught by the references cited by the Examiner.

Leymann et al. discloses a method of extended transaction processing. In one embodiment (Fig. 5), messages are moved between a first queue 505 (Q1) and a second queue 506 (Q2) in a transaction 502 (T2) that is part of a chained transaction. The Examiner refers to the passage at column 7, lines 19-31, which states:

> [A] transaction, also called a transactional application, comprises a sequence of operations that changes recoverable resources and data such as a database from one consistent state into another. A transaction processing system guarantees that if a transaction executes some updates against recoverable resources and data, and if a failure occurs before the transaction reaches its normal termination or an interim point of consistency, then those updates will be undone (roll-back). When a new point of consistency has been reached and all updates made by the transaction must be made permanent, the transaction is committed.

The Examiner equates this with applicants' claimed step of defining a list with each list entry key corresponding to an uncommitted message falling within an uncommitted key range and each list entry key corresponding to a committed message falling within a committed key range. Applicants respectfully disagree. While Leymann et al. relates to message queuing and transaction processing, it has little else in common with applicants' claimed system. Thus, while this passage discusses committing transactions, it does not discuss any particular data structure for storing transaction elements, much less a list structure with keys defining committed and uncommitted portions. Therefore, it fails to teach this element of applicants' claimed combination.

The Examiner also refers to the passage at column 11, lines 29-48, which states:

> When chaining within a member transaction Ti the next member transaction Tj this processing request is inserted, as part of the commit scope of the calling transaction Ti, into a request queue. This request queue is of course a recoverable, i.e. durable, resource and the insertion operation fulfills the ACIDicity requirements. A multitude of these recoverable request queues thus support persistent recoverable messaging. Taking the processing request of the next member transaction Tj out of the queue is part of the

> commit scope of Tj.
>
> If the process of chaining within CT from a member transaction Ti the next member transaction Tj is asynchronous in nature, these asynchronous transaction processing requests do not result in a processing of the next member transaction Tj at once. Instead the processing request is inserted, as part of the commit scope of the calling transaction Ti, into a request queue. It is left to the TP system when this processing request will result in the execution of the requested next member transaction Tj.

The Examiner equates this with applicants' claimed step of adding to the list a list entry having a list entry key within the uncommitted key range in response to a request to write a message to the queue. Again, applicants respectfully disagree. In applicants' claimed system, the committed status of a list entry is indicated by its list entry key. The entry is committed if the key falls within a committed key range. While the cited passage of Leymann et al. speaks of a request queue, the reference does not teach a keyed list structure, nor does it teach writing an entry to an uncommitted portion of such a list structure as claimed by applicants.

Nor does the other cited reference, Denny et al, cure the deficiencies of Leymann et al. Denny et al. discloses a method of handling protected conversation messages across IMS restart in a shared queues environment. Denny et al. describes how unit-of-work elements (UOWEs) from a request list are moved between a staging queue and a ready queue, among other operations. As described at column 8, lines 28-49:

> The above request list is then processed element-by-element, see FIG. 2, block 2.03. In the next step, each UOWE is analyzed. If the status of the commit indicator corresponding to a given UOWE is in doubt, then the first data object corresponding to that UOWE is put to the staging queue, see FIG. 2, blocks 2.04 and 2.05. If the UOWE attributes indicate that the message is ready to be put in to the ready queue, then the first data object corresponding to that UOWE is put to the ready queue, see FIG. 2, blocks 2.06 and 2.07. If the UOWE attributes indicate that it is ready to be released, the system releases the UOWE, and corresponding entry is removed from the request list, see FIG. 2,

blocks 2.08 and 2.09. If the UOWE contains the commit indicator, the system moves all data objects under the same commit scope designated by the recovery token into the ready queue and scans UOWEs using the recovery token, see FIG. 2, blocks 2.15-19. If any UOWEs with matching recovery tokens are found, they are also put into the ready queue. Finally, if the UOWE contains the abort indicator, the system deletes all data objects under the same commit scope, which is determined by the recovery token, see FIG. 2, blocks 2.10-14.

The Examiner equates this with applicants' claimed step of adding to the list a list entry having a list entry key within the uncommitted key range in response to a request to write a message to the queue. The Examiner concludes that it would be obvious from this "to incorporate inputting write messages to the uncommitted list" (paper no. 5, page 4). Again, applicants respectfully disagree.

While Denny et al., like Leymann et al., relates to message queuing and transaction processing, it likewise has little else in common with applicants' claimed system. Applicants do not just write message to an uncommitted list, as the Examiner apparently argues, but writes such messages to the uncommitted portion of a list as defined by list entry keys. This is nowhere taught in the references cited.

Conclusion

For the foregoing reasons, applicants respectfully submit that claims 1-26 distinguish patentably over the art cited by the Examiner. Applicants therefore respectfully request reconsideration of the application as amended and, upon such consideration, a favorable action on the merits. Such action is earnestly solicited..

Respectfully submitted,
DAVID A. ELKO

By [illegible]
William A. Kinnaman, Jr.
Registration No. 27,650
Phone: (845) 433-1175
Fax: (845) 432-9601

WAK/wak